Submission in Response to NSF CI 2030 Request for Information

DATE AND TIME: 2017-04-05 11:29:13
REFERENCE NO: 231

Author Names & Affiliations

- Jia Rao - The University of Texas at Arlington
- Hong Jiang - The University of Texas at Arlington
- Song Jiang - The University of Texas at Arlington
- Hao Che - The University of Texas at Arlington

Contact Email Address (for NSF use only)

(Hidden)

Research Domain, discipline, and sub-discipline

Computer Science and Engineering; Computer Systems and Architecture, Parallel and Distributed Computing; Cloud Computing

Title of Submission

Towards Open Programmable Cyberinfrastructure via Serverless Cloud Computing

Abstract (maximum ~200 words).

The advances of data-driven science have made cyberinfrastructure a critical platform for collecting, processing, and analyzing massive amounts of social, biomedical, commercial, and physical data. The complexity of big data, including its sheer volume, high velocity, and large variety, poses challenges to building future cyberinfrastructure for effective and efficient data analytics. We believe that open programmability is the key to enabling data-driven innovations in different research fields. However, making cyberinfrastructure, which comprises heterogeneous devices such as massively parallel servers, mobile devices, sensors, and specialized accelerators, programmable and interoperable is non-trivial. We envision that cyberinfrastructure design should center on a container-based serverless computing model. We identify a few advancements that are needed to accomplish this goal. These include flexible and on-demand data abstraction in distributed storage, lightweight and ephemeral container overlay network, composable data analytic algorithms, and precise QoS management in container-based systems.

Question 1 Research Challenge(s) (maximum ~1200 words): Describe current or emerging science or engineering research challenge(s), providing context in terms of recent research activities and standing questions in the field.

The longtime pursuit of utility computing has led to the development of many service-oriented systems, such as grid and cloud computing systems. These systems offer on-demand, well-defined services from shared infrastructures that are usually built from massively dense and multi-core servers. While commercially successful, exemplified by leading cloud services such as Amazon Web Services (AWS), Google Compute Cloud, and Microsoft Azure, many challenges arise facing service-oriented computing to support emerging Data Science

applications. Different from traditional data mining, data science aims to discover actionable knowledge from geographically distributed, heterogeneous, and vast amount of social, biomedical, commercial, and physical data. Its applications include but are not limited to social behavior and network analysis, imaging-based medical diagnosis, brain-inspired computing, and self-driving vehicles. These applications pose unique challenges to the software, hardware, and data infrastructures of the underlying computing systems.

• Sheer volume of data. Recent success of artificial intelligence (AI) and big data analytics relies on the access to vast amounts of high-quality data. For example, the training of a deep convolutional neural network requires millions of samples for accurate image recognition. Furthermore, each sample alone can contain a rich set of data; a histopathology image of a tissue specimen at 40x magnification contains 100,000 X 100,000 pixels in resolution and close to 30GB in size. The storage and transmission of such large data sets require a non-trivial amount of space and network bandwidth. The difficult dilemma facing researchers is that although the movement of data to remote service providers incurs considerable cost and causes the data lock-in problem, it is unfortunately necessary. Oftentimes, meaningful computation on such data is only possible when the data set is in the proximity of computational resources. Moreover, some data is ephemeral and appears in the form of continuous streams, which needs to be processed in real time.

• Heterogeneous and dispersed data sources. The sources of big data vary from field to field and can come from multiple geographically dispersed and heterogeneous repositories/devices. Typical data sources include archives (e.g., scanned documents), data warehouses (e.g., databases and Hadoop/Spark), media files (images and videos), public web (e.g., social network), machine logs (e.g., application logs), and sensors (e.g., medical devices, GPS, household appliances). These data sources vary in volume, velocity, variety, and veracity, and have distinct computational requirements. Researchers in data science often need to fuse data from multiple sources to derive multifaceted views on a problem or combine data from real-time acquisition with historical data. The heterogeneity of various data types and their physically dispersed locations require considerable preprocessing or even data movement before presented to researchers for analysis.

• Computation complexity and heterogeneity. Not only do different data sources differ in format, but the computation on them is also heterogeneous. While some computations are simple and last for a short period of time, e.g., performing a sub-second query from well-structured databases, other computations can be quite complex and time-consuming, e.g., multi-hour training of a deep neural network. The computation can also be inherently heterogeneous and executes on distinct devices. For example, data-parallel applications with high arithmetic intensity should run on GPUs while computations on distributed data sets with a rich set of control flows should execute on many-core CPUs or a cluster of machines. Emerging computing modes, such as edge computing and IOT, considered integral parts of the cloud ecosystem, often require specialized accelerators, in addition to GPUs, (e.g., ASIC, FPGA, Automata Processors, etc.).

• Lack of programmability. The existing cloud has a wide spectrum of service models, ranging from software-as-a-service (SaaS), platform-as-a-service (PaaS), to infrastructure-as-a-service (IaaS). SaaS provides on-demand access to commercial software; PaaS allows users to develop customized software on a programmable platform without worrying about the underlying server maintenance and deployment, such as scalability and reliability; IaaS offers the most flexibility and presents users with virtualized hardware abstractions, on top of which operating systems (OSes) and server software can be installed. However, the existing service models do not fit well in the data science domain, in which computations performed on multiple locations/sources can be heterogeneous. There lacks a generic programmable architecture with good usability to develop data science applications across various types of devices. While the software and platform abstractions in SaaS and PaaS are unlikely to meet every user's need, the hardware abstraction in IaaS places significant burdens on average users to develop and deploy their own applications. For example, it is non-trivial and time-consuming to write a distributed data analytic program that runs on a cluster of heterogeneous devices. Ideally, the cloud service should provide OS and data abstractions that are similar to processes/threads and files in traditional systems, abstract resources on multiple locations, and are portable across different platforms.

• Demand for high efficiency, fine-grained resource accounting and provisioning, and precise QoS management. Due to the computational complexity and the sheer volume of data, many data science applications demand a considerable amount of computing resources and storage space. Unfortunately, such resources may not be readily available to individual researchers at their labs or host institutions. Viable solutions include outsourcing the computation to external cloud services or consolidating underutilized resources across departments/labs within an institution. Both require that resources be efficiently utilized to reduce the monetary cost and to support more users. In multi-tenant systems, fine-grained resource accounting and precise quality-of-service (QoS) management are keys to high efficiency. Multi-dimensional resources, e.g., CPU cycles, memory, storage, and accelerator devices, should be accurately allocated and charged to individual users to manage inefficiency caused by resource multiplexing. Furthermore, QoS metrics should be properly defined and their objectives should be strictly enforced. As such, the shared hardware infrastructure can be properly provisioned at the capacity planning phase to avoid under- or over-provisioning. Most importantly, clear and precise definitions of fairness and differentiation help attain high resource utilization and guarantee QoS objectives.

• Security and privacy concerns. Moving data to external clouds or sharing it among departments on campus may not be possible if security and privacy issues on sensitive data are not addressed. For example, restrictions on access to patient medical records and individual-level genomic data complicate the implementation of data science applications. Proper data protection and security control protocols should be designed and evaluated by multiple committees. This is a tedious procedure and hampers the development of new applications, especially collaborative projects that require building new data infrastructures across multiple repositories. To address these issues, data science applications should avoid moving data if in-situ computation is possible. Additionally, the computation on sensitive data should be composable and individually verifiable to comply with various privacy and security requirements at different sites.

Question 2 Cyberinfrastructure Needed to Address the Research Challenge(s) (maximum ~1200 words): Describe any limitations or absence of existing cyberinfrastructure, and/or specific technical advancements in cyberinfrastructure (e.g. advanced computing, data infrastructure, software infrastructure, applications, networking, cybersecurity), that must be addressed to accomplish the identified research challenge(s).

We envision that cyberinfrastructure that has an integrated software and hardware design for data and resource virtualization is the answer to the aforementioned challenges facing data science. Different from the 2012 CIF21 vision, which aimed to exploit massively parallel computers and highly distributed systems to build hybrid cyberinfrastructure, we believe that future cyberinfrastructure will be a federation of heterogeneous computing devices and data repositories, each managed by an autonomous organization. The cyberinfrastructure is loosely coupled in structure but presents users with unified data and resource abstractions. For example, data stored on multiple repositories can be abstracted as generic key-value pairs and be looked up from centralized or distributed indexes. Computations can be abstracted as pre-defined, composable and reusable functions and/or library routines deployed on each device. MapReduce is one such programming model widely adopted in distributed systems. However, future cyberinfrastructure should go beyond MapReduce's single program multiple data (SPMD) programming model and allow heterogeneous programs to run on multiple data and on heterogeneous computing substrates (HPMD). To achieve high efficiency, especially on edge devices powered by battery e.g., sensors, resources should be allocated at fine granularity and in an on-demand manner.

To enable the aforementioned programming model, we believe future cyberinfrastructure will be built on microservices or serverless computing. Both are based on feather-like container-based cloud computing. Container technology allows a service to be deployed in milliseconds across various platforms, e.g., Linux, Windows, high-volume servers, mobile devices, and sensors. In what follows, we describe a use case for serverless computing in cyberinfrastructure, discuss state-of-the-art practice in the cloud industry, and outline the challenges to be addressed in container-based serverless computing.

Use case: multiple institutions maintain their repositories of machine learning models at separate sites. A researcher can use the machine learning repositories in several ways – 1) use the stored models for prediction on local data sets; 2) update and customize existing models using new data sets; 3) develop complicated predictive analytics based on an ensemble of models across different repositories. Researchers will write analytic programs in a way similar to that on shared-memory machines except that the program comprises modular "functions", which can be implemented and deployed on individual devices, instead of statements. The functions take globally addressable data abstractions, key-value pairs, as input, and are orchestrated by a high-level programming runtime to form a semantically meaningful program. The execution of such a program involves invocations of functions, each represented by a container, on multiple machines/devices. The cyberinfrastructure launches an ephemeral cluster to handle the execution flow of the program. The nodes of the computing cluster are lightweight containers and the cluster is destroyed after program execution completes. This new programming model supports a variety of analytic jobs, from one-shot ad hoc queries that last a few seconds to long-running model training tasks. The outcome of the analytic program, e.g., an updated model, can be persisted to the repositories for future use.

State-of-the-art industry practice. Leading cloud providers have launched a new cloud service model, function-as-a-service (FaaS), to support microservices or serverless computing. Examples include Amazon AWS Lambda, Google Cloud Function, and Microsoft Azure microservices. FaaS allows users to deploy pre-defined functions on the cloud, which can be scheduled for execution or triggered by events. FaaS also allows for a new pricing model that only charges for the period during which user code is running. While similar in concept to the programming model we envisioned, FaaS is not readily applicable to future cyberinfrastructure. FaaS assumes that data is stored on centralized cloud storage, e.g., Amazon S3, and requires that data collected at edge devices be uploaded to the cloud before it can be processed. In cyberinfrastructure, data movement can be quite expensive and sometimes not possible due to data protection. As a result, computation needs to be moved to data as deployable functions encapsulated in containers in a data-centric fashion.
Below are the advancements needed in cyberinfrastructure to support the container-based serverless programming model.

• Data virtualization. While data is physically dispersed on multiple locations, there should be a unified view of data presented to users to ease programming. The characteristics of cyberinfrastracture present several challenges to storage and data abstraction. First, since only the data involved in the analytic program is needed for computation, it is not necessary to make all data in multiple repositories visible to one particular user. This requires on-demand data abstraction upon the request of a microservice. Second, cyberinfrastructure comprises a large number of heterogeneous devices, some connected by wireless networks with intermittent connectivity. Traditional data abstractions, such as distributed file systems, can be quite expensive if data consistency is a requirement in such a dynamic environment. Finally, the data abstraction should be lightweight and facilitate data sharing. In earlier discussions, a researcher may need to incrementally update an existing model while other users are still using the old model. The data abstraction should support versioning stored models via snapshots. To reduce storage redundancy, cyberinfrastructure can explore copy-on-write techniques but any optimizations on space efficiency should not add much overhead to the on-demand creation of a data abstraction as most workloads are short-lived.

• Container networks. Although cyberinfrastucture aims to provide a programming interface similar to shared-memory machines, the underlying implementation is inevitably an IP network. To isolate analytic programs, each program should be placed into a virtual cluster of containers with its own private network. While a cyberinfrastructure may contain tens or hundreds of machines or devices, the container network can be much larger. As a container usually encapsulates a single-purpose function, its footprint is comparable to processes in traditional OSes. Therefore, there can be hundreds or thousands of containers on each machine. On-demand creation of an overlay network to enclose thousands of containers, which only exist for a few seconds, poses great challenges to container networking. We perform a survey on existing container overlays, such as weave, flannel vxlan, calico, and NAT, and the results show significant overhead. As cyberinfrastructue should span heterogeneous devices, the networking infrastructure will likely be based on network function virtualization. Efficiently supporting hundreds or thousands of short-lived overlay networks may require new algorithms or engineering efforts on maintaining the overlays.

• Composable data analytics. To fully exploit the benefits of cyberinfrastructure, including virtually infinite scalability, versatility, and efficiency, data science applications should be made composable. This requires algorithmic advances in some fields, e.g., deep learning, to break the original computation into loosely-coupled, but composable components, each running in a container.

• Multi-tenancy support. The ultimate goal of a cyberinfrastructure will be a versatile platform that meets diverse needs of different research. Thus, multi-tenancy must be supported and should have higher requirements than traditional systems. First, since computation can be heterogeneous, fairness and differentiation should be extended to consider multi-dimensional resources. Second, resource accounting, provisioning, and scheduling should be performed at fine granularity. Being statistically fair in long-term resource allocation is not adequate for short-lived user requests. Third, while modern operating systems, e.g., Linux, support resource accounting, isolation, and allocation for a collection of processes, e.g., Linux control groups (cgroups), it is challenging to extend group-based resource management to multiple machines.